Visualization and Analysis of Student Interactions in an Adaptive Exploratory Learning Environment

Dror Ben-Naim, Nadine Marcus, Mike Bain

School of Computer Science and Engineering
University of New South Wales, Sydney, Australia

Abstract. In this paper we describe research that applies educational data-visualization and data mining techniques in an Adaptive Exploratory eLearning Environment called the Adaptive eLearning Platform. Using a novel visualization tool called the Solution Trace Graph, we were able to visualize student interactions and thus gain insights that led to the refinement of the intelligently adapted remediation in the system. An important observation we make concerns the employment of a software design methodology which we refer to as Virtual Apparatus Framework (VAF). By using VAF to develop eLearning content, the process of developing intelligently adapted remediation in an exploratory learning scenario, and subsequently the analysis of students' behaviour, is greatly enhanced and simplified.

Key words: Adaptive e-Learning, Intelligent Tutoring Systems, Educational Data-Mining, Virtual Apparatus Framework, Exploratory Learning Environments

Introduction

Exploratory Learning Environments (ELE) have recently emerged as an alternative to controlled and step-guided educational systems. ELE typically emphasise learning by interaction and exploration of the environment via its interface.

In Exploratory Learning Environments, the discovery and self guided nature of the activity plays a role in increasing students' motivation, which may contribute towards learning [1].

ELE are suitable for educational activities that involve simulations, where learners can experiment with different aspects and parameters of a given phenomenon to observe the effects these changes have on outcomes of the simulation. These types of environments can often feature goal-free learning, which has been empirically demonstrated by Sweller and colleagues to lead to improved learning [2]. Unconstrained exploration of a large problem-space may overwhelm learners in terms of number of choices (see [2]) therefore, we suggest that effective learning in ELE can be achieved using guided exploration.

In ELE, keeping track of learners' interaction for the purpose of analysis of student's behaviour is difficult, as typically, these environments provide a rich user interface for the student to explore. Monitoring students' interactions and logging their activity yields a high-bandwidth data set – a large set of attribute-values, per each interaction. Making sense of this data, in order to improve the activity is a difficult task and is subject to research efforts by many [3].

This paper focuses on the Adaptive eLearning Platform (AeLP) - a distributed software architecture for the development and deployment of Adaptive eLearning content. It has been argued by a number of researchers working on Adaptive eLearning technologies that distributed, component-based architecture is the way to introduce these technologies into the mainstream [4-6].

The motivation behind the Adaptive eLearning Platform is a desire to create an Intelligent Virtual Exploratory Learning Environment. Research to date [7, 8] suggests that students benefit from an interactive learning environment in which they can have some control over their learning experience. Teachers also benefit from the ability to track the students' progress during a learning activity.

Virtual Apparatus Framework for Designing and Developing eLearning Activities

Virtual apparatus framework for educational content development was described as a content development paradigm that can promote reusability of Learning Objects (LO) and reduce the effort required in developing educational content [9, 10]. With Virtual Apparatus Framework we can approach the eLearning content development process in the same way we approach developing (real world) teaching laboratory activities. In the design of a lab activity the teacher is responsible for setting up the experiment table and composing the experiment notes. The apparatus is built by third party companies. VAF employs a similar separation of concerns: separation of content and presentation. The development of the presentational software components (called Virtual Apparatus) is delegated to the software engineers while the educational (content) aspects of the activities are authored by the teacher. VAF adheres to the Model View Control (MVC) design pattern in software engineering. In MVC, data, its presentation and actions are all kept separate [11]. Essentially, VAF is an attempt to increase separation of concerns in educational software engineering. It's similar to other design patterns that support componentization of the educational building blocks that have been suggested in the past [12].

One important software engineering implication that stems from the choice of VAF as a design pattern is that the Virtual Apparatus is developed such that its properties, and its subcomponents' properties, can be *SET* and *GET* by an external application through an Application Programming Interface (API). With VA's API in place, authoring educational activities can be thought of as analogous to real lab content authoring. In the real world, teachers set up an experiment and compose notes explaining to students how to interact with the apparatus in order to achieve some educational goal. During lab activity, demonstrators (intelligently) give students remediation based on their problems and *the state their apparatus is in*. Similarly,

developing content in VAF, we import VA into a virtual "experiment table" and compose notes that instruct students through the activity. The Adaptive eLearning Platform addresses how we define the correct answer or more specifically, how we define the correct state the Virtual Apparatus should be brought to? And how do we intelligently remediate students based on their problems?

The Adaptive eLearning Platform

The AeLP was originally designed to complement the laboratory component for first year physics courses at the University of New South Wales (UNSW), by providing virtual experiments and laboratory-like activities. Since 2004, first year physics teaching at UNSW was reformed to incorporate "Exploratorial" sessions which are described in [13]. Exploratorials aim to integrate observations, experiments, calculations and theory, to offer the insight of a lecture and the student-led analysis of a tutorial and the hands-on measurement of a lab.

Adaptation (in computer science) is an overly broad term that characterises any software systems that can change some features based on some User Model. In eLearning, adaptation is the subject of much research and development. Some researchers are concerned with adaptive navigational support of education hypermedia [14]. Other groups focus on adapting sequences of higher granularity content such as lessons, syllabi and courses (e.g.[15]). The AeLP's adaptivity focuses on adapted remediation within an interaction task and adapted sequencing of low granularity content such as a questions (for further discussion see [9]).

Authoring Adaptive Tutorials

Authoring an Adaptive Tutorial (AT) using the AeLP Author, consists of importing prefabricated Virtual Apparatus into the system, setting it to some initial state (via its API), and then composing notes that'll direct the students through their interaction in the environment. For each VA imported, we compose a set of questions that require interaction and exploration of different forms from students (e.g. "set the apparatus to such condition to maximize parameter x"). Using the AeLP Author, the teacher defines the correct state the apparatus should be brought to (again, in terms of its API), and possible error states. Those states are called trap-states and are defined by conjunctions and disjunctions of conditions. To each trap-state, a remediation (feedback, textual or not) is attached. The problem's state-space is defined as the set of all possible states the VA can get into.

Conditions are defined as triples of [targetName, operator, value]. For example a condition might be:

```
VA.angleControl.value = 0
```

Where the targetName is "VA.angleControl.value", the operator is '=' and the value is '0'.

The available targetNames for defining conditions come from the VA's API. When a VA is imported into the platform the platform creates an Object Model interface to the VA's properties that it can access [Figure 1].

The underlying assumption behind developing content using this paradigm is that any question can be modelled as a task to bring some system from an initial state, defined by some initial conditions, to some other correct state, defined by another set of conditions, by interacting with it. A more formal definition for a question can be:

> Let question Q, be defined as the process by which a state machine M is manipulated by student S to change its internal state from initial state I to correct state C. A transition between I and C must exist.

When we develop the content this way, questions can be considered mini expert systems. Rules in the AeLP are fired when the VA is in a particular state defined by its set of conditions. Antecedents are defined as states the VA can get into: points or regions in the state-space of the system. The state-space is vast if not infinite (e.g. a state can be defined by a rule such that the student landed in a particular, other state, twice). The question is then: how do you define all the possible states you want to remediate? Using the AeLP author the teacher actually only needs to define some states, those that support his hypothesis regarding what mistakes the students will make, and what type of remediation is needed.

In our experience, VAF was found to be useful for the development of virtual experiments and interaction based activities that are based on simulations and are thus exploratory in nature.

Evaluation at the University of New South Wales

The Adaptive eLearning Platform is currently being deployed across three schools at UNSW: Physics, Mechanical Engineering and Computer Science. The AeLP's adaptive content serves 600 students a year. For example, in the School of Physics, we developed an AT for teaching first year physics students the concept of Faraday's Law [Figure 1]. The students were given a virtual coil rotating in magnetic field, a virtual oscilloscope and were required to complete various tasks, such as maximizing the magnetic flux, or calculating the electromotive force generated by the rotation of the coil.

Data-Mining the AeLP Logs

The Adaptive Tutorial's logs saved in the system contain massive amounts of information. We store a complete snapshot of the entire system per each student's interaction. Each snapshot contains values of all the inspectable properties in Virtual Apparatus and other runtime information. In this way we keep a complete log of what students have done.

There are several motivations for our data analysis:

1. We want to know how students interacted with the VA; in what ways they manipulated the VA's states
2. We want to know if our hypotheses about expected user problems fit student's behaviour
3. We'd like to get feedback on how effective our adapted predefined remediation was
4. In case our remediation was not effective, we would like to be able to drilldown and understand why
5. We would like to investigate what students' misconceptions and mistakes we did not anticipate when authoring the Adaptive Tutorial
6. We want to be able to pick up abnormalities in the system's behaviour, and possibly identify problems we did not anticipate in the authoring phase
7. We would like to be able to classify students based on their performance and add this information to their student model

With these motivations in mind, we designed and developed the *Adaptive Tutorial Analyser* [Figure 2] component of the AeLP.

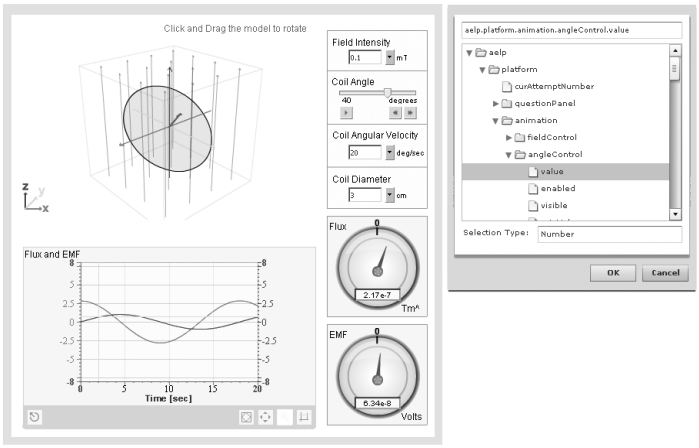

Figure.1. A Virtual Apparatus Setup (left) and its corresponding API (right) in the AeLP Author. Using the API the teacher defines the init, correct and error trap-states.

Data-Visualization Strategy

As the students attempt to solve a task, they manipulate the UI controls of the VA thus changing its state. During the authoring phase, we define trap-states on a question's available state-space. These represent the states we think the students might erroneously bring the VA into. We are thus interested in understanding the way students navigate through this state-space in order to arrive at the correct state. Essentially we can think of the process a student takes in order to solve the task as a trace through the problem's state-space. Our first attempt with the Adaptive Tutorial

Analyser was to try and visualize this process. For this purpose, we developed the Solution Trace Graph [Figure 2]. The Solution Trace Graph (STG) shows student progression per question as a transition between the question's states. A student's *solution trace* can be visualized as a "multicolumn" graph, where each column contains all question trap-states as nodes. There are as many columns in the STG as the total number of attempts the student needed in order to get to the correct state. An edge in the graph is a transition between two states. The graph's horizontal direction represents progression in time measured as discreet solution attempts.

To investigate multiple students' behaviour, we superimpose their solution traces on one STG. The number of columns in the STG is now the maximum number of attempts it took any student to arrive at the correct state. In a superimposed STG, an edge's weight is defined as the number of students that have passed between two trap-states between the same attempt numbers (e.g. moving between state A to B on the second solution attempt) Edge Weights are noted by a label on the edge line, and are also colour coded.

Inspecting the data in such a manner gives an immediate visual intuition about students' behaviour during the Adaptive Tutorial.

Assessing the Effectiveness of Adaptive Remediation

In the STG, the teacher can see a breakdown of how many students arrive at each state (weights on incoming edges to a state-node) and, based on the remediation given in this state, how many students moved to the correct state (outgoing edge leading to 'correct' state). Of particular interest are edges that connect the same states across different columns. Such transitions imply either that the remediation given to the students was ineffective - because the student "landed" on the same error state - or that the error state was too general and students landed there given a broad set of circumstances. Once we identify that our remediation was not effective, the next step is to drilldown and to inspect what it is that the student has done, that we did not anticipate. In essence, we want to know why our remediation wasn't successful in assisting students and we do that by inspecting what they have done after the remediation was given. To enable this, we need a way to look at all the relevant edges coming out of the desired state, and understand what is common between them, i.e. what was the behaviour we missed when creating the adaptive remediation?

This is achieved by two means: teacher driven and data driven analysis. The teacher driven approach enables the teacher to investigate patterns in students' behaviour. The data driven approach uses various data-mining algorithms in order to pick patterns in the data, but will not be discussed in this paper.

To enable teacher insight into the data, we organize the snapshot data in an easily inspectable DataGrid [Figure 2]. Clicking on an edge, the STG populates the snapshot DataGrid with the information corresponding to the interactions that resulted in the transition between the edge's states.

Each column in the DataGrid represents a dimension in the state-space (an inspectable property of the VA or the runtime environment). Columns can be sorted by clicking on their header. It is, therefore, easy to notice values of certain properties that are recurring. In the example shown in Figure 2 we noticed that 3 students made

the mistake of omitting the exponent part of the solution (instead of entering 1.13e-3 they entered 1.13). Although technically a mistake, we think this type of error should be remediated in an adaptive way, and possibly incur a small score penalty. Using the AeLP author an extra error trap-state can be created based on the condition:

```
questionPanel.userInput == "1.13"
```

and the adapted remediation can be for e.g.:

```
"Almost correct, have you got the significant figures
right?"
```

In this sense, the process of developing an Adaptive Tutorial is through a continual refinement of the rule-base. In this way we are avoiding the expert system "knowledge acquisition bottleneck" problem.

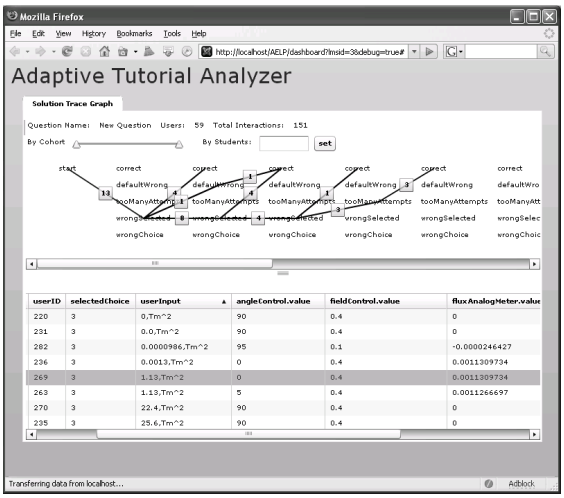

Figure. 2. The Solution Trace Graph shows superimposed solution traces for a sub group of students. Starting from the left, 13 students 'landed' in the '*wrongSelected*' state. After being given the adaptive remediation, 4 students were able to get to the 'correct' state whilst 8 students continued to '*wrongSelected*'. This suggests that the remediation might need to be revised. The DataGrid is populated with the snapshot information representing the state the Virtual Apparatus was in when the student attempted a solution. The grid allows for quick inspection of mistake patterns. In this case we see students entered "1.13" instead of "1.13e-3".

Specifying an Overly General Trap-State

An important feature that the STG allows us to investigate is the edges that lead to the '*defaultWrong*' trap-state. The *defaultWrong* trap-state is basically there to give generic feedback to students if they have done something that we did not anticipate. This state does not have any condition attached to it, and therefore its (empty) rule

fires when all the other custom trap-states are not applicable. A sufficiently adaptive activity should have no edges leading to the *defaultWrong* trap-state because all possible errors were adaptively remediated for. However, in reality we sometimes count on the *defaultWrong* remediation where the question/task at hand does not involve manipulation of the apparatus (e.g. questions such as: multiple choice, free input, true false) and we can count on the per-attempt custom feedback feature to progressively expose students to the answer. In open tasks of either guided or exploratory nature, the edges that lead to the *defaultWrong* are investigated in order to understand what type of mistakes the students have made, and how we can adaptively remediate for them.

Pattern Recognition

Recognizing what behaviour patterns are emerging in the data is the process by which the teacher identifies what kind of mistakes the students are making.

One interesting pattern that the STG can easily reveal is state transitions that are predictors of other state transitions. The STG can show us, for instance, that 90% of students who landed in a trap-state A, continued, after being remediated to trap-state B. A visual way to explore this is to project the entire STG to just two columns.

Data-mining could also be applied to discover these patterns automatically, possibly as conditional probabilities in a Bayes Net or association rules.

Identifying Bugs and Abnormalities

As the horizontal dimension of the STG represents solution attempts, a long STG means students were unsuccessful in getting to the correct state after a large number of attempts. Investigating interactions leading to long STG's, we occasionally found that these resulted from bugs in the Virtual Apparatus, or the User's system. Such feedback is really useful with production environments.

Future work

Through our experience developing the STG, some interesting possibilities became immediately apparent:

- Replaying Student's Activities: The DataGrid contains snapshot information. Using this information we can recreate the system's state. In effect, we can replay the student's activity. Teachers using our system expressed the desire to "just click and be able to replay the activity the way the students did it", thus better enabling understanding what has happened. Extending the Adaptive Tutorial Analyser by developing the Activity player is a short-term goal of our R&D team.
- Live Monitoring: in order to support supervised, live learning scenarios, we are investigating a Live-STG that is continuously updated with real-time data.

- Labelling sequences: an interesting consequence of visually analysing data using the STG is that it is possible to identify sub groups of users who behave in a similar manner and label them for the purpose of user and group modelling. This possibility will be investigated in the future.
- Collaboration – We are developing live-collaboration support into the AeLP so that students can work on a VA simultaneously in groups of up to 5 students. Data mining the group work is a challenge we are currently working on.

Conclusion

This paper presents a novel approach to support adaptivity in an exploratory learning environment – the Solution Trace Graph (STG) - a tool that visualizes student's behaviour and provides visual feedback about the effectiveness of adaptive remediation.

The STG models student behaviour in an exploratory environment, as a solution trace between predefined trap-states. We found when using the STG to analyse interaction data in the Adaptive eLearning Platform, teachers could clearly gain valuable information about how students interacted with the exploratory environment in a visual format, therefore, get feedback regarding the quality of their Adaptive Tutorials. More importantly, by identifying what errors were not accounted for in the adaptive remediation, and adding further trap-states into the Adaptive Tutorial, the process of refining the educational activity becomes iterative and gradual. This greatly eases the deployment of adaptive content as it facilitates the analysis and refinement process. In this way, the AeLP addresses what is known as the "knowledge acquisition bottleneck" problem of developing expert systems. Moreover, based on the analysis of past performance, educational content can more easily be developed to meet student's needs. The personalised feedback of an individual teacher can also be simulated.

We note that one precondition to all of this functionality is our definition of eLearning activities as transitions between question states. The strength of this paradigm stems from the fact that this definition is extremely flexible and can be applied to most eLearning activities.

In this sense authoring an adaptive activity can be an incremental process that is both teacher driven and data driven. Using the STG, the AeLP helps teachers enhance their teaching practice, allowing them to develop content that can be customised to students' needs.

References:

1. Reiser, B., W. Copen, M. Ranney, A. Hamid, and D. Kimberg, *Cognitive and Motivational Consequences of Tutoring and Discovery Learning*, in *Technical Report*. 1994, The Institute for the Learning Sciences.
2. Sweller, J., *Instructional Design In Technical Areas*. 1999: ACER Press.

3. Cristóbal Romero, M.P., Toon Calders, Silvia R. Viola, *International Workshop on Applying Data Mining in e-Learning (ADML'07)*, in *Second European Conference on Technology Enhanced Learning (EC-TEL07)*. 2007, CEUR-WS.org: Crete, Greece.
4. Brusilovsky, P., S. Sosnovsky, and O. Shcherbinina, *User Modeling in a Distributed E-Learning Architecture*, in *User Modeling 2005*. 2005. p. 387-391.
5. Carmona, C. and R. Conejo, *A Learner Model in a Distributed Environment*, in *Adaptive Hypermedia and Adaptive Web-Based Systems*. 2004. p. 353-359.
6. Conlan, O., et al. *An Architecture for integrating Adaptive Hypermedia Services with Open Learning Environments*. in *World Conference on Educational Multimedia, Hypermedia and Telecommunications 2002*. 2002. Denver, Colorado, USA: AACE.
7. Mayer, R.E., Chandler, P., *When learning is just a click away: Does simple user interaction foster deeper understanding of multimedia messages.* Journal of Educational Psychology, 2001. 93: p. 390-397.
8. Betrancourt, M., *The Animation and Interactivity Principles in Multimedia Learning*, in *The Cambridge Handbook of Multimedia learning*, R.E. Mayer, Editor. 2005, Cambridge University Press: Cambridge. p. 278-296.
9. Dror Ben-Naim, NadineMarcus, Michael Bain. *Virtual Apparatus Framework Approach to Constructing Adaptive Tutorials*. in *The 2007 International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government*. 2007. Las Vegas, Nevada, USW: CSREA Press.
10. Albert Ip, R.C., *A Model for Authoring Virtual Experiments in Web-based Courses*, in *Conference Australasian Society for Computers in Learning in Tertiary Education* 1996: Adelaide, Australia.
11. Reenskaug, T., *The Model-View-Controller (MVC) Its Past and Present*, in *JavaZONE*. 2003, JAOO, Århus: Oslo, Norway.
12. Ch. Kynigos, M.K., Th. Hadzilacos, *Mathematics with Component Oriented Exploratory Software*. International Journal of Computers in Mathematics Education, 1998.
13. McAlpine i., C.M., Newbury R., Hatsidimitris G. *Multimedia Enhanced Active Learning Studio: a development in physics teaching*. in *Proceedings of ED-MEDIA 2005 - World Conference on Educational Multimedia, Hypermedia and Telecommunications*. 2005. Montreal, Canada.
14. Brusilovsky, P., *Adaptive Navigation Support*, in *The Adaptive Web*. 2007. p. 263-290.
15. Santos, O.C., C. Barrera, and J.G. Boticario, *An Overview of aLFanet: An Adaptive iLMS Based on Standards*, in *Adaptive Hypermedia and Adaptive Web-Based Systems*. 2004. p. 429-432.